UNIVERSIDAD NACIONAL
SAN AGUSTIN

FACULTAD DE PRODUCCION Y SERVICIOS

Informe sobre R Tree

Estructuras de Datos Avanzadas

LUIS GUILLERMO VILLANUEVA FLORES

AREQUIPA
2019

1. Introducción

Los árboles-R o R-árboles son estructuras de datos de tipo árbol similares a los árboles-B, con la diferencia de que se utilizan para métodos de acceso espacial, es decir, para indexar información multidimensional; por ejemplo, las coordenadas (x, y) de un lugar geográfico. Un problema con aplicación práctica en el mundo real podría ser: "Encontrar todos los museos en un radio de dos kilómetros alrededor de la posición actual".
La estructura de datos divide el espacio de forma jerárquica en conjuntos, posiblemente superpuestos.

2. Estructura

Cada nodo de un árbol-R tiene un número variable de entradas (hasta un máximo predefinido). Cada entrada de un nodo interno almacena dos datos: una forma de identificar a un nodo hijo y el conjunto límite de todas las entradas de ese nodo hijo.
Los algoritmos de inserción y borrado utilizan los conjuntos límite de los nodos para asegurar que elementos cercanos están localizados en la misma hoja (en particular, un nuevo elemento será insertado en la hoja que requiera el menor aumento del conjunto límite). Cada entrada de una hoja contiene dos datos: una forma de identificar el elemento actual (que, alternativamente, podría estar directamente en el nodo) y el conjunto límite de ese elemento.

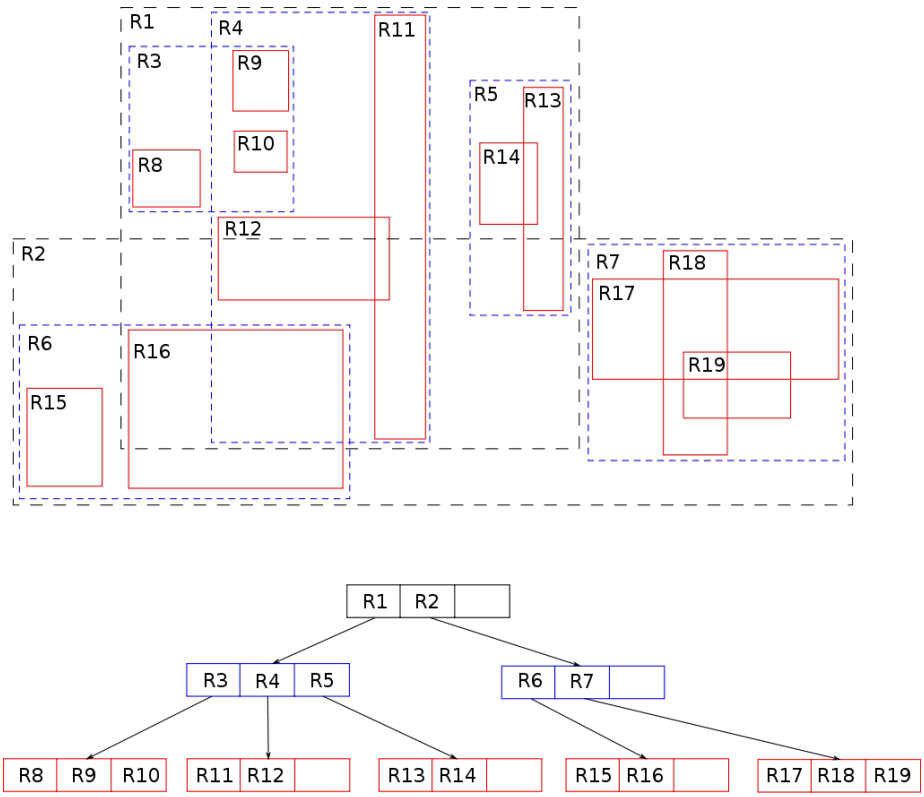

Figura 1: Representación del R tree con un máximo de 3

2.1. Representación en 3D

Así como hay la representación mostrada anteriormente pero en 2D también existe para dimensión 3 y así sucesivamente, puesto que solo se puede observar hasta 3 dimensiones , se muestra un gráfico donde se representa la estrucutura pero en 3 dimensiones.

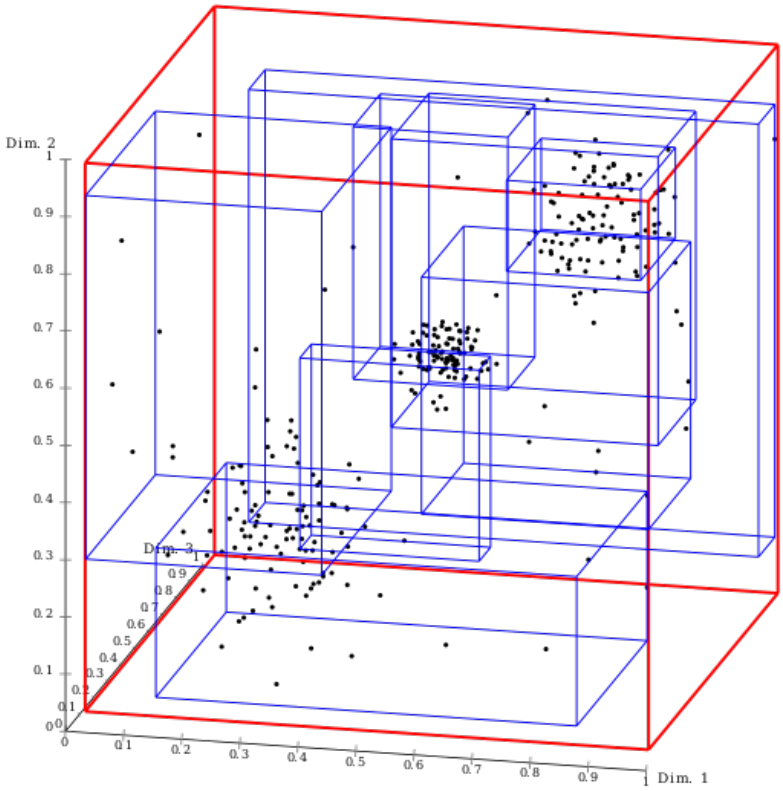

Figura 2: Representación del R tree en 3 dimensiones

3. Clases usadas

He usado las clases point, para representar a un punto, también he usado la clase mbr, la clase nodo y mi clase principal la clase rtree.A continuacón se muestra cada una con sus métodos correspondientes.

```
class point
{
public:
  int x,y;
  point(int a,int b){
    x=a;
    y=b;
  }
};
```

Listing 1: Clase Point

```
class mbr
{
public:
  int maxi;
  int perimeter;

  int largo;
  int ancho;
  mbr( int m){

    maxi=m;
  }

  int getperim(){
    return perimeter;
  }

  void calperim(point izinf,point dersup){
    largo=abs(izinf.y-dersup.y);
    ancho=abs(izinf.x-dersup.x);

    perimeter= largo*ancho*2;
  }

  void updateper(vector<point> b){
    int c=calperim(b[0],b[1]);
    for(int i=0;i<b.size();i++){
      for(int j=0;j<b.size();j++){
        if(c==0)continue;
        else if(calperim(b[i],b[j])<c){
          c=calperim(b[i],b[j]);
        }

      }

    }
    perimeter=c;


```

```
  }
};
```

Listing 2: Clase MBR

```
class nodo
{

public:
  vector<point>datos;
  vector<mbr>nodes;
  bool isleaf;
  nodo(){
    isleaf=true;
  }

};
```

Listing 3: Clase Nodo

```
class rtree
{

public:
  mbr M;
  int max;
  nodo* root;
  rtree(int num){
    max=num;
    root=new nodo();
  }

  mbr choose_subtree(nodo *u,point p){
    mbr c(this->max);
    for(int i=0;i<u->nodes.size();i++){
      if(u->nodes[i]->calperim(u->datos[i],p)<u->nodes[i+1]->
          calperim(u->datos[i+1],p)){
        c=u->nodes[i];
      }

    }
    return c;


  }
  void insert(nodo *u, point p){
    if(u->isleaf){
      u->datos.push_back(p);
      if(this.max<u->datos.size()){
        handle_overflow(u);
      }
    }

    else{
      nodo *v=new nodo();
      v=choose_subtree(u,p);
      insert(v,p);
    }
```

```
38
39  }
40  void handle_overflow(nodo *u){
41    if(u->isleaf)
42      splitL(u);
43    else
44      split(u);
45    if(this->root==u){
46      nodo* aux=new nodo();
47      aux=u;
48      aux->isleaf=false;
49      root->nodes=u;
50
51    }
52    else{
53      nodo* aux=new nodo();
54      aux->nodes=u;
55      updateper(aux->nodes);
56      if(this->max<aux->nodes.size()){
57        handle_overflow(aux);
58      }
59
60    }
61  }
62
63
64
65  void splitL(nodo *u){
66    vector<nodo>a;
67    vector<nodo>b;
68    int m=u->datos.size();
69    sort(u->datos.begin(),u->datos.end(),sortPx);
70    for (int i =ceil(0.4*this->max) ; i < m-ceil(0.4*this->max); ++
          i)
71    {
72      a.datos.push_back(u->datos[i]);
73    }
74
75    for(int i=0;i<this->u->datos.size();i++){
76      for(int j=0;j<a.size();j++){
77        if(a.datos[i]!=u->datos[i]){
78          b.datos.push_back(u->datos[i]);
79        }
80      }
81    }
82  }
83
84  void split(nodo *u){
85    vector<nodo>a;
86    vector<nodo>b;
87    int m=u->datos.size();
88    sort(u->datos.begin(),u->datos.end(),sortPx);
89    for (int i =ceil(0.4*this->max) ; i < m-ceil(0.4*this->max); ++
          i)
90    {
91      a.nodes.push_back(u->nodes[i]);
92    }
```

```
93
94      for(int i=0;i<this->u->datos.size();i++){
95        for(int j=0;j<a.size();j++){
96          if(a.nodes[i]!=u->nodes[i]){
97            b.nodes.push_back(u->nodes[i]);
98          }
99        }
100     }
101   }
102
103 };
```

Listing 4: Clase rtree

Emplee una función adicional para ordenar los puntos, y se muestra a continuación:

```
1  bool sortPx(point a1, point a2) {
2    return a1.x < a2.x;
3  }
4  bool sortPy(point a1, point a2) {
5    return a1.y < a2.y;
6  }
```

Listing 5: Funciones para ordenar

4. Conclusiones

- Pude observar que al igual que en el Quadtree y Octree esta estructura nos puede ayudar mucho al momento de querer distribuir datos de una manera no uniforme, ya que pueden estar dispersos en cualquier espacio donde nosotros lo definamos y querramos que esten.
- Al igual que todas las estructuras anteriores que hemos visto, me di cuenta que es una manera eficiente de representar datos de manera espacial ya sea en 2d o en n-dimensiones.
- Además de ser una estructura de datos espaciales nos pede ayudar a encontrar el perímetro más pequeño entre unos puntos y eso es muy funcdamental al momento de querer hacer consultas o distribuir datos de manera eficiente, puesto que sabemos en que rango consultar simplemente y sabemos en que región se puede encontrar una cantidad de puntos.

5. Referencias

- Árbol-R. (2019). Retrieved 18 December 2019, from https://es.wikipedia.org/wiki/
- R-tree - R-tree - qwertyu.wiki. (2019). Retrieved 18 December 2019, from https://es.qwertyu.wiki/wiki/R-tree
- Introduction to R-tree - GeeksforGeeks. (2019). Retrieved 18 December 2019, from https://www.geeksforgeeks.org/introduction-to-r-tree/